# 6指针实验

## 6.1 实验目的

熟练掌握指针的说明、赋值、使用。通过掌握用指针引用数组的元素，熟悉指向数组的指针的使用。熟练掌握字符数组与字符串的使用，同时熟练运用指针数组及字符指针数组。掌握指针函数与函数指针的用法。运用带有参数的main函数。

## 6.2 必做题

### 6.2.1 源程序改错

**【题目】**

下面的源程序中是否存在错误？如果存在，原因是什么？如果存在错误，要求在计算机上对这个源程序进行调试修改，使之能够正确执行。

#include "stdio.h"

int main(void)

{

float \*p;

scanf("%f",p);

printf("%f\n",\*p);

}

**【错误原因分析及改错方案】**

源程序中没有对悬挂指针进行赋值，所以需要定义一个变量，使指针p指向它，再对这个变量赋值，所以改为 float a，\*p=&a；

**【修改后代码】**

#include "stdio.h"

int main(void)

{

float a,\*p=&a;

scanf("%f",p);

printf("%f\n",\*p);

return 0;

**}**

**【测试数据、程序测试理论结果及测试结果】**

本程序非常简单，仅设计一组测试即可：输入5，预计输出5.000000，实际输出如图，完全符合预测。

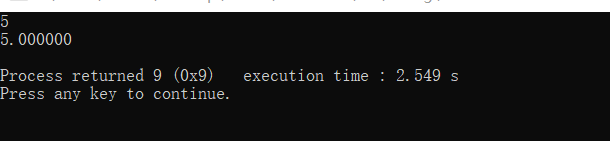


图6-1 实验六必做题6.2.1 测试结果

### 6.2.2 源程序修改替换

**【题目】**

（1）下面的程序通过函数指针和菜单选择来调用字符串拷贝函数或字符串连接函数，请在下划线处填写合适的表达式、语句、或代码片段来完善该程序。

（2）为了使程序不受scanf、getchar、gets等函数输入后回车符的影响，请修改第（1）题程序，按要求输出下面结果：（（输入）表示该数据是键盘输入数据）

1 copy string.

2 connect string.

3 exit.

input a number (1-3) please!

2 （输入）

input the first string please!

the more you learn, （输入）

input the second string please!

the more you get. （输入）

the result is the more you learn,the more you get.

#include "stdio.h"

#include "string.h"

int main(void)

{

char a[80],b[80],c[160],\*result=c;

int choice,i;

do{

printf("\t\t1 copy string.\n");

printf("\t\t2 connect string.\n");

printf("\t\t3 exit.\n");

printf("\t\tinput a number (1-3) please!\n");

scanf("%d",&choice);

}while(choice<1 || choice>5);

switch(choice){

case 1:

p=strcpy;

break;

case 2:

p=strcat;

break;

case 3:

goto down;

}

getchar();

printf("input the first string please!\n");

i=0;

printf("input the second string please!\n");

i=0;

result= (a,b);

printf("the result is %s\n",result);

down:

;

}

**【修改后代码】**

#include "stdio.h"

#include "string.h"

int main(void)

{

char \*(\*p)(char[],const char[]);

char a[80],b[80],c[160],\*result=c;

int choice,i;

do{

printf("\t\t1 copy string.\n");

printf("\t\t2 connect string.\n");

printf("\t\t3 exit.\n");

printf("\t\tinput a number (1-3) please!\n");

scanf("%d",&choice);

}

while(choice<1 || choice>5);

switch(choice){

case 1:

p=strcpy;

break;

case 2:

p=strcat;

break;

case 3:

goto down;

}

getchar();

printf("input the first string please!\n");

i=0;

gets(a);

printf("input the second string please!\n");

i=0;

gets(b);

result=(\*p)(a,b);

printf("the result is %s\n",result);

down:

return 0;

}

**【测试数据、测试理论结果及测试结果】**

（1）本测试包含两种函数，每种函数设一组测试数据，同时包含输入异常的数据，最终测试退出，因此，共3组测试数据：

输入1，第一个字符hello，第二个字符world！，将第二个字符复制到第一个字符并覆盖，结果为world；

实际测试结果如图所示，完全符合要求。

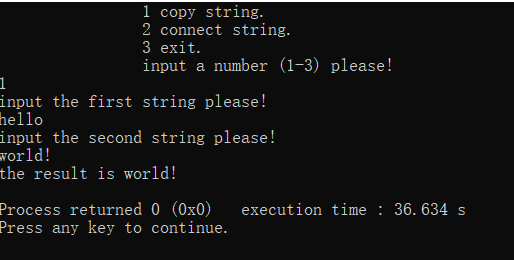


图6-2 实验六必做题6.2.2（1）样例1测试结果

### 6.2.3 跟踪调试

**【题目】**

请按下面的要求对所给源程序进行操作，并回答问题和排除错误。

(1) 单步执行源程序。进入strcpy时,watch窗口中s为何值？返回main时, watch窗口中s为何值？

(2) 排除源程序中的错误，使程序输出结果为： there is a boat on the lake.

#include "stdio.h"

char \*strcpy(char \*,char \*);

void main(void)

{

char a[20],b[60]="there is a boat on the lake.";

printf("%s\n",strcpy(a,b));

}

char \*strcpy(char \*s,char \*t)

{

while(\*s++=\*t++)

;

return (s);

}

**【调试过程】**

调试过程共设置两个断点，分别在进入函数strcpy和返回main处。命中第一个断点时，读取局部变量窗口中的s的值，为0x012ffecc，指向的地址所储存的字符串无效,如图6-5所示；命中第二个断点时，读取局部变量窗口中的s的值，为0x012ffec9，指向的地址所储存的字符串无效,如图6-6所示。

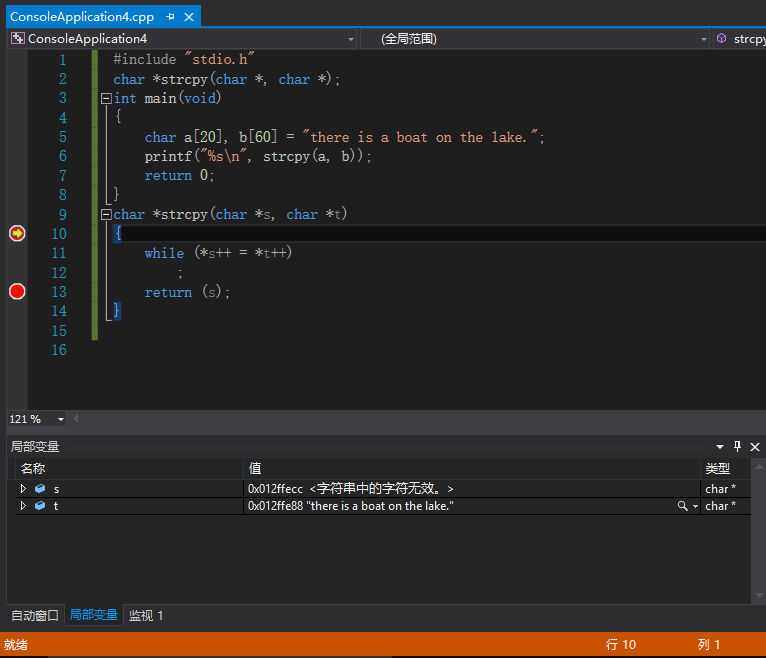


图6-5 实验六必做题6.2.3跟踪调试（1）

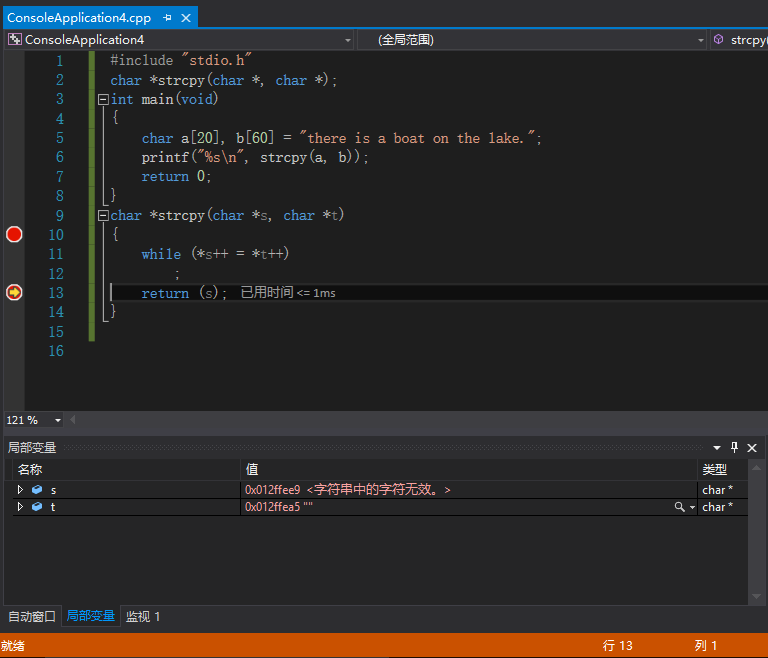


图6-6 实验六必做题6.2.3跟踪调试（2）

### 6.2.4 十六进制输出

**【题目】**

已知一个长整型变量占4个字节，其中每个字节又分成高4位和低4位。试编写一个程序,从该长整型变量的高字节开始，依次取出每个字节的高4位和低4位并以十六进制数字字符的形式进行显示。

**【算法流程图】**

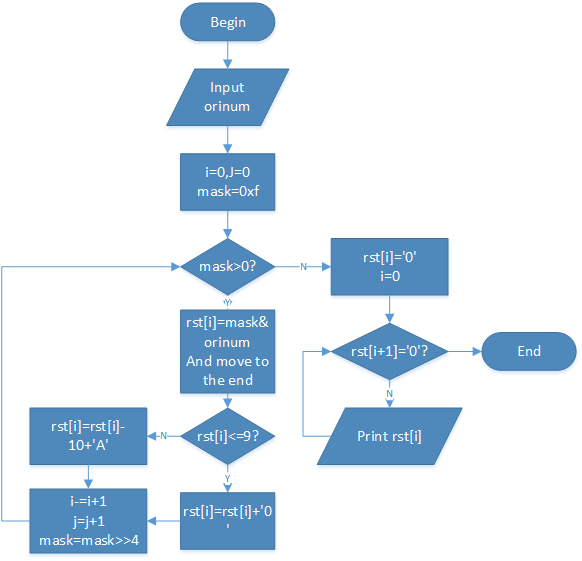


图6-7 实验六必做题6.2.4算法流程图

**【程序清单】**

#include<stdio.h>

#include<stdlib.h>

int main()

{

unsigned long mask = 0xflu << (8 \* sizeof(long) - 4);

char \*rst = (char\*)malloc(sizeof(char) \* 100);

long \*orinum = (long\*)malloc(sizeof(long) \* 100);

int i = 0, j, k;

mask = 0xflu << (8 \* sizeof(long) - 4);

for (i = 0, j = 0; mask > 0; mask >>= 4, j++, i++)

{

//依次取出每一位

\*(rst + i) = (char)((mask&(\*(orinum+k))) >> (sizeof(long) \* 8 - 4 - 4 \* j));

if (\*(rst + i) <= 9)

\*(rst + i) += '0';

else if (rst[i] > 9)

\*(rst + i) = \*(rst + i) - 10 + 'A';

}

\*(rst + i) = '\0';//最后赋值结束符\0

for (i = 0; \*(rst + i + 1) != '\0'; i++)

printf("%c ", \*(rst + i));

printf("%c\n", \*(rst + i));

}

return 0;

}

**【测试样例，测试理论结果及测试结果】**

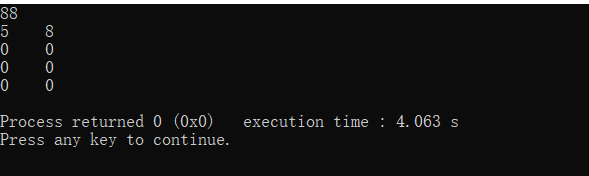


图6-8 实验六必做题6.2.4测试结果

### 6.2.5 去除空格空行

**【题目】**

利用大小为n的指针数组指向用gets函数输入的n行，每行不超过80个字符。编写一个函数，它将每一行中连续的多个空格字符压缩为一个空格字符。在调用函数中输出压缩空格后的各行，空行不予输出。

**【算法】**

首先确定你需要输入的行数，然后分别用gets函数输入字符串，并且定义一个指针数组以用来调用二维字符数组，然后检查每个字符串的空格，将多余空格合并之后，就按照顺序输出，空行不予输出。

**【程序清单】**

#include<stdio.h>

#include<string.h>

#define max 81

int check(char \*s);

int main()

{

int n,x,i,j=0;

scanf("%d",&n);

char a[n][max],\*p[n];

getchar();//吸收换行符

for(i=0;i<n;i++){

p[i]=a[i];//将二维数组的各个一级数组的首地址存入指针数组中

gets(p[i]);

}

printf("print:\n");

while(j<n)

{

x=check(p[j]);//检查数组并将其按要求将多余空格合并。

if(x){

printf("%s\n",p[j]);

j++;

}

else j++;//若此行为换行，则跳过。

}

getch();

return 0;

}

int check(char \*s)

{

int i=0,k=0,y=0,flag=0;

if(\*s!='\0')//是否有输入字符。

{

y=1;

while(\*s!='\0')

{

if(\*s==' ')

{

if(flag!=1)//若为第一个空格则跳过压缩数组

flag=1;

else while(\*(s+i-1)!='\0')//压缩字符数组。使空格变为一个

{

\*(s+i-1)=\*(s+i);

i++,k=1;

}

if(!k) s++;//多余空格压缩完毕就跳到下个字符。

i=0,k=0;

}

else//不是空格。检查通过直接下个字符。

{

s++;

flag=0;

}

}

}

return y;

}

**【测试结果】**

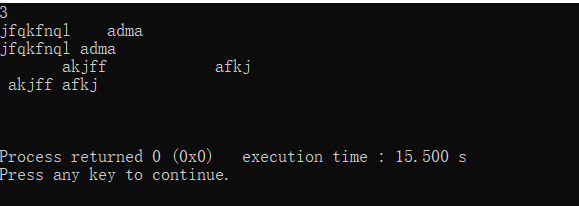


图6-9 实验六必做题6.2.5测试结果

### 6.2.6 命令行参数

**【题目】**

编写一个程序,输入n个整数，排序后输出。排序的原则由命令行可选参数-d决定，有参数-d时按递减顺序排序，否则按递增顺序排序。要求将排序算法定义成函数，利用指向函数的指针使该函数实现递增或递减排序。

**【程序代码】**

#include<stdio.h>

int main(int argc, int argv[])

{

int i, j, n;

int num[50];

scanf("%d", &n);

for (i = 0; i < n; i++)

scanf("%d", &num[i]);

for (j = n - 1; j > 1; j--)

for (i = 1; i <= j; i++)

{

//升序

if (num[i - 1] > num[i])

{

int tmp = num[i - 1];

num[i - 1] = num[i];

num[i] = tmp;

}

//降序

if (num[i - 1] < num[i] && argc == 2)

{

int tmp = num[i - 1];

num[i - 1] = num[i];

num[i] = tmp;

}

}

for (i = 0; i < n; i++)

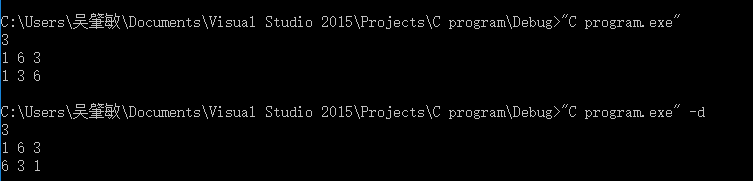
printf("%d ", num[i]);

printf("\n");

}

**【测试数据、测试理论结果及测试结果】**

测试数据分为两组，含-d与不含-d，在cmd中进入.exe所在文件夹，先输入"C program.exe"，然后测试输入3，数据为1、6、3，应按升序排序，即结果为1 3 6，然后输入相同的数据，应按降序排序，结果为6 3 1。实际输出结果如图6-10所示，符合预测。



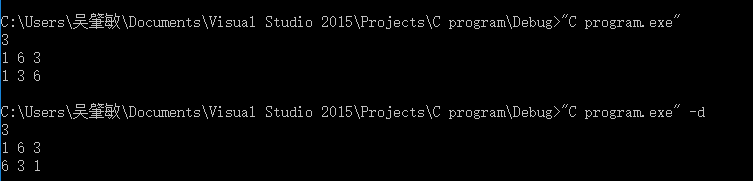


图6-10 实验六 6.2.6 测试结果

### 6.2.7 成绩处理

**【题目】**

设某个班有N个学生，每个学生修了M门课程（用#define定义N、M）。输入M门课程的名称，然后依次输入N个学生中每个学生所修M门课程的成绩并且都存放到相应的数组中。试编写下列函数：

(a) 计算每个学生各门课程平均成绩；

(b) 计算全班每门课程的平均成绩；

(c) 分别统计低于全班各门课程平均成绩的人数；

(d) 分别统计全班各门课程不及格的人数和90分以上（含90分）的人数。

在调用函数中输出上面各函数的计算结果。（要求都用指针操作，不得使用下标操作。）

**【算法流程图】**

本程序主要分为1、输入数据，2、计算每个人平均分，3、计算每个课程平均分，4、统计每个课程低于平均分的人数，5、统计每个课程不及格的人数，统计高于90分人数。本程序算法为简单的查找算法，重点考察指针表示数组。且由于3、4、5算法基本相同，故以3为例。具体程序框图如图~图。

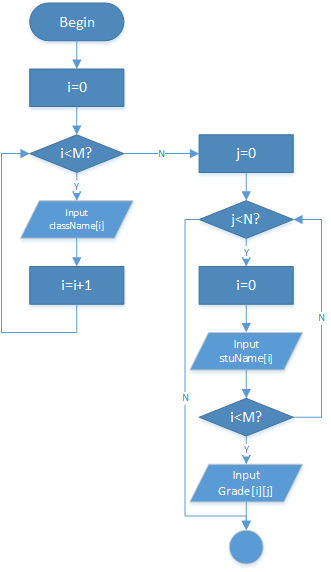


图6-11 实验六必做题6.2.7 算法流程图（1）

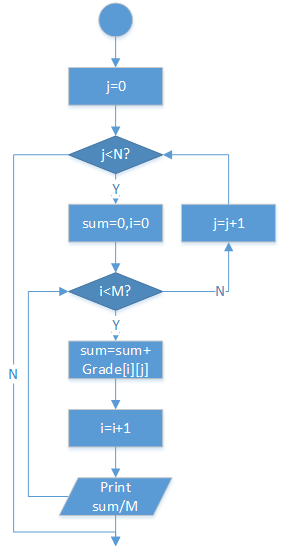


图6-12 实验六必做题6.2.7 算法流程图（2）

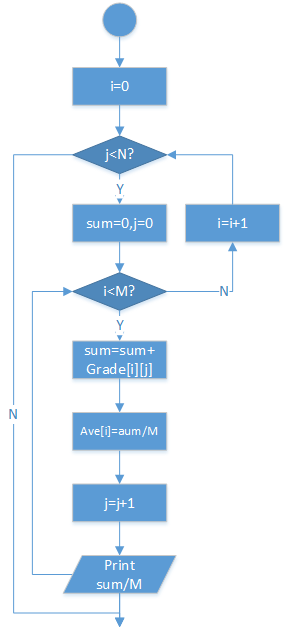


图6-13 实验六必做题6.2.7 算法流程图（3）

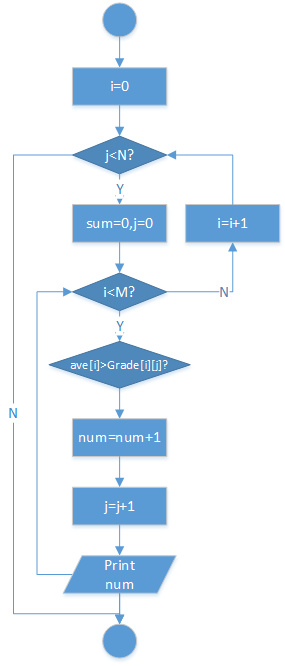


图6-14 实验六必做题6.2.7 算法流程图（4）

**【代码清单】**

#include<stdio.h>

#include<stdlib.h>

#define M 5 //number of classes

#define N 5 //number of students

int main()

{

char \*className = (char\*)malloc(8\*sizeof(char)\*M);

char \*\*studentName = (char\*\*)malloc(8\*sizeof(char\*)\*N);

float \*\*studentGrade = (float\*\*)malloc(8\*sizeof(float\*)\*N);

int i, j;

//输入数据

printf("Input 5 classes' name (single letters only):\n");

for (i = 0; i < M; i++)

{

scanf("%c", className + i);

getchar();

}

printf("Please input each student's name followed by \nhis (or her) scores of classes:\n");

for (j = 0; j < N; j++)

{

\*(studentName + j) = (char\*)malloc(8 \* sizeof(char) \* 100);

\*(studentGrade + j) = (float\*)malloc(8 \* sizeof(float)\*M);

scanf("%s", \*(studentName + j));

for (i = 0; i < M; i++)

scanf("%f", \*(studentGrade + j) + i);

}

//计算每个学生平均数并输出

float sum;

for (j = 0; j < N; j++)

{

sum = 0;

for (i = 0; i < M; i++)

sum += \*(\*(studentGrade + j) + i);

printf("Average score of %s is %.2f\n", \*(studentName + j), sum / M);

}

//计算每个课程的平均数并输出

float \*ave = (float\*)malloc(8 \* sizeof(float)\*M);

for (i = 0; i < N; i++)

{

sum = 0;

for (j = 0; j < M; j++)

sum += \*(\*(studentGrade + j) + i);

\*(ave + i) = sum / M;

printf("Average score of %c is %.2f\n", \*(className + i), sum / M);

}

//统计低于全班各课程成绩平均的人数

for (i = 0; i < N; i++)

{

int num = 0;

for (j = 0; j < M; j++)

if (\*(ave + i) > \*(\*(studentGrade + j) + i))

num++;

printf("Number of students lower than avg of %c is %d\n", \*(className + i), num);

}

//统计全班不及格人数

for (i = 0; i < N; i++)

{

int num = 0;

for (j = 0; j < M; j++)

if (\*(\*(studentGrade + j) + i) < 60.0)

num++;

printf("Number of students fail %c is %d\n", \*(className + i), num);

}

//统计全班90分以上人数

for (i = 0; i < N; i++)

{

int num = 0;

for (j = 0; j < M; j++)

if (\*(\*(studentGrade + j) + i) > 90.0)

num++;

printf("Number of students %c perfect is %d\n", \*(className + i), num);

}

return 0;

}

**【测试数据及测试理论结果】**

本实验情况较为单一，仅设计一组数据即可，设计师尽量加大数据的混乱度。因此，设计以下输入：

1、首先输入课程名：A B C D E；

2、然后输入每个学生的姓名（为一个单词，不含空格）和成绩：

Qw

78 94 58.5 74.8 91.4

Er

59.9 88.3 66 76 92.3

Ty

77 56.9 74 88.1 81.3

Ui

65 85.4 70 79.4 88.7

Op

70 89.4 85.5 79.3 87.5

预测输出没人平均数如下：

Average score of Qw is 79.34

Average score of Er is 76.50

Average score of Ty is 75.46

Average score of Ui is 77.70

Average score of Op is 82.34

预测输出课程平均分如下：

Average score of A is 69.98

Average score of B is 82.80

Average score of C is 70.80

Average score of D is 79.52

Average score of E is 88.24

预测输出统计低于平均分人数如下：

Number of students lower than avg of A is 2

Number of students lower than avg of B is 1

Number of students lower than avg of C is 3

Number of students lower than avg of D is 4

Number of students lower than avg of E is 2

预测输出不及格人数如下：

Number of students fail A is 1

Number of students fail B is 1

Number of students fail C is 1

Number of students fail D is 0

Number of students fail E is 0

预测输出90分以上人数如下：

Number of students A perfect is 0

Number of students B perfect is 1

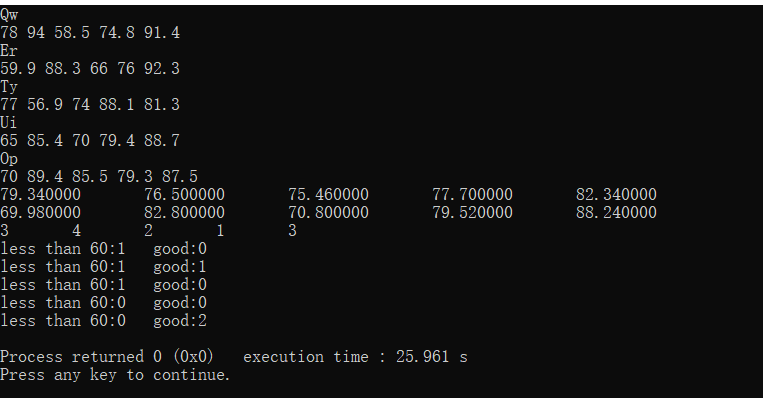
Number of students C perfect is 0

Number of students D perfect is 0

Number of students E perfect is 2

**实际输出如图6-15、图6-16所示，完全符合预测。**

**【测试结果】**



6.3 选做题

### 6.3.1 小数相加

**【题目】**

设有N位整数和M位小数（N=20，M=10）的数据a,b。编程计算a+b并输出结果。

**【算法流程图】**

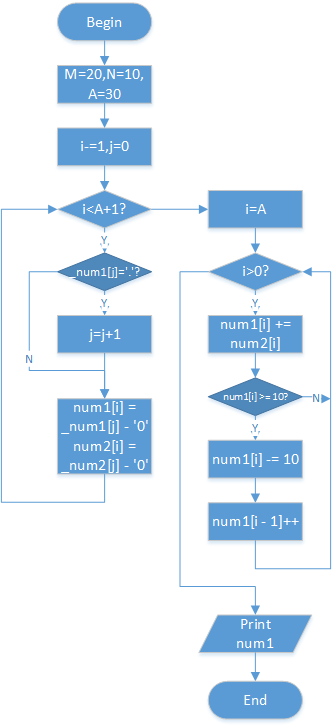


图6-17 实验六 选做题6.3.1算法流程图

**【程序清单】**

#include<stdio.h>

#include<string.h>

#define M 20

#define N 10

#define A (M+N)

int main()

{

char \_num1[A + 2], \_num2[A + 2];

printf("Please input two numbers with (20nums).(10nums):\n");

scanf("%s", \_num1);

scanf("%s", \_num2);

int num1[A + 1] = { 0 }, num2[A + 1] = { 0 };

int i, j, flag = 0;

for (i = 1, j = 0; i < A + 1; i++, j++)

{

if (\_num1[j] == '.') j++;

num1[i] = \_num1[j] - '0';

num2[i] = \_num2[j] - '0';

}

for (i = A; i > 0; i--)

{

num1[i] += num2[i];

if (num1[i] >= 10)

{

num1[i] -= 10;

num1[i - 1]++;

}

}

if (num1[0] != 0)

printf("%d", num1[0]);

for (i = 1; i <= A; i++)

{

if (i == M + 1)

printf(".");

printf("%d", num1[i]);

}

printf("\n");

return 0;

}

**【测试数据、测试理论结果及测试结果】**

根据首位为0、末位为0以及引发进位设计如下三组数据。

表6-2 实验六 选做题6.3.1测试数据、测试理论结果及测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 样例 | 测试数据 | 测试理论结果 | 测试结果 |
| （1） | 01111111110111111111.0111111111  +  01111111110111111111.0111111111 | 0222222222  0222222222  .0222222222 | 如图6-18 |
| （2） | 12345678901234567890.1234567890  +  12345678901234567890.1234567890 | 2469135780  2469135780  .2469135780 | 如图6-19 |
| （3） | 65874654712647846587.2168745612  +  84658723547565358465.5622148754 | 15053337826  0213205052  .7790894366 | 如图6-20 |

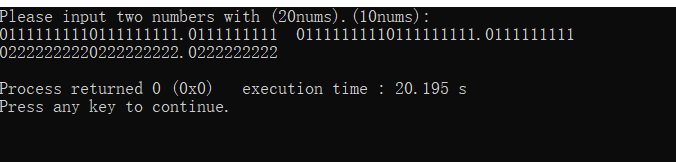


图6-18 实验六 选做题6.3.1样例（1）测试结果

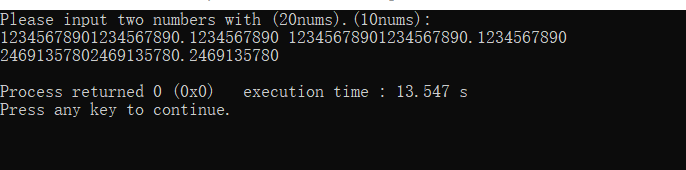


图6-19 实验六 选做题6.3.1样例（2）测试结果

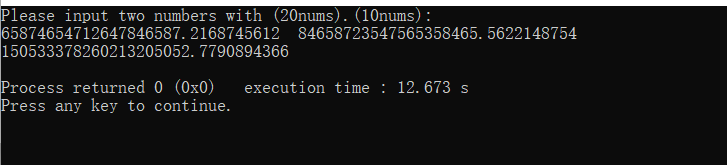


图6-20 实验六 选做题6.3.1样例（3）测试结果

### 6.3.2 复杂声明

**【题目】**

编写使用复杂声明char \*(\*p[2])(const char \*,const char \*);的程序。

提示：p中元素可为strcmp、strstr等函数名。

**【算法】**

其思想是跟源程序修改替换中程序修改替换的思想一样，不过就是这个是用的是一个复杂声明，声明了一个存放函数指针的数组，并且将两个字符函数的指针赋给它，相当于给这个函数起了一个别的名称来执行功能，输入两个字符串，按照输入条件来判断用其中一个函数指针，即执行被指函数的功能。

**【程序清单】**

#include<stdio.h>

#include<string.h>

char \*\_strcat(const char \*, const char \*);

char \*\_strcpy(const char \*, const char \*);

char c[100];

int main()

{

int x=1;

char s[162],a[81],\*q=s;

char \*(\*p[2])(const char\*,const char \*);

p[0] = \_strcat;

p[1] = \_strcpy;

printf("输入两行字符串:\n");

gets(s);

gets(a);

printf("输入你想要的指令:\n");

printf("1-将两个字符串连接并输出\n");

printf("2-用第二行字符串覆盖第一行字符串并输出\n0-退出\n");

while(x!=0)

{

scanf("%d",&x);

if(x==1){

q=p[0](s,a);

printf("%s\n",q);

}

else if(x==2){

q=p[1](s,a);

printf("%s\n",q);

}

}

printf("\nPress any key to quit...");

getchar();

return 0;

}

char \*\_strcat(const char \*a, const char \*b)

{

strcpy(c, a);

strcat(c, b);

return c;

}

char \*\_strcpy(const char \*a, const char \*b)

{

strcpy(c, a);

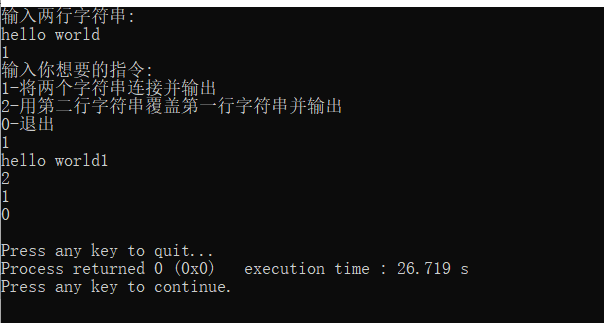
return strcpy(c, b);

}

**【测试数据、测试理论结果及测试结果】**

普通样例，Hello world；！； 输出结果预测，选择命令1的结果为Hello world！选择命令2的输出结果为！；实际输出结果如图6-21。

图6-21 实验六选做题6.3.2测试结果



## 6.4 自设题

**6.4.1 验证数组**

**【题目】**

编写一个程序来验证p是不是数组。

**【原理】**

如果p是数组，则p的值和&p的值是相同的；

p++的值会因为p是数组而发生改变。

**【程序清单】**

#include<stdio.h>

void sub(int p[])

{

printf("the size of space p:%d\n",sizeof(p));

printf("the result of p:%p\n",p);

printf("the address of p:%p\n",&p);

p++;

printf("the result of p++:%p\n",p);

}

int main()

{

int a[10]={1,3,5,2,7,9,6,8,0,4};

printf("the size of space a:%d\n",sizeof(a));

printf("the result of a:%p\n",a);

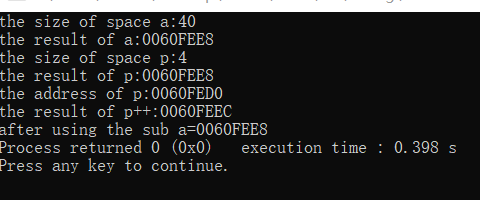
sub(a);

printf("after using the sub a=%p",a);

return 0;

}

**【输出结果】**



**【自设小结】**

通过完成这个自设题，对数组和指针有了深刻的理解，包括数组的首地址与数组名的关系。

## 6.5 小结

源程序改错中，依赖于平时良好的输入习惯，在输入过程中已经可以检查出大多数语法错误，或者拼写错误。接下来再检查一下逻辑是否有误，确认最终无误即可。

源程序修改与替换题锻炼我们将充分利用指针的能力，同时要注意指针的含义，指针指向对象的含义，充分理解防止出现问题，也为以后多重引用奠定基础。

程序设计题则是自主的设计实践，根据题目要求自主设计指针，增强对于所学知识的熟练度和理解程度，锻炼编写代码的能力以及并通过问题的进一步引申锻炼思维的灵活性。